Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Інститут телекомунікацій, радіоелектроніки та електронної техніки



Звіт до лабораторної роботи №10

з дисципліни «Програмування, частина 1»

Виконав:

ст. гр. АП-11

Білоніжко Віталій

Викладач: Чайковський І. Б

Львів-2022

Мета роботи: Дослідити властивості циклічних операторів мови С.

1.

/\* Celsius and Fahrengeit \*/

/\* C=(5/9)\*(F-32) \*/

#include <stdio.h>

int main()

{

int fahr, celsius;

int lower, upper, step;

lower = 0;

upper = 300;

step = 20;

fahr = lower;

printf("\n\nCelsius Fahrengeit\n");

while (fahr <= upper)

{

celsius = 5 \* (fahr - 32) / 9;

printf("%10d\t%8d\n", fahr, celsius);

fahr = fahr + step;

}

return 0;

}

2.

// Здійснити зміну типу обчислюваних величин на float та відлагодити нову версію програми.

/\* Celsius and Fahrengeit \*/

/\* C=(5/9)\*(F-32) \*/

#include <stdio.h>

int main()

{

float fahr, celsius;

int lower, upper, step;

lower = 0;

upper = 300;

step = 20;

fahr = lower;

printf("\n\nCelsius Fahrengeit\n");

while (fahr <= upper)

{

celsius = 5.0 \* (fahr - 32.0) / 9.0;

printf("%10.0f\t%8.1f\n", fahr, celsius);

fahr = fahr + step;

}

return 0;

}

3.

// 3. Змодифікувати оператор циклу на do, відлагодити нову версію.

/\* Celsius and Fahrengeit \*/

/\* C=(5/9)\*(F-32) \*/

#include <stdio.h>

int main()

{

float fahr, celsius;

int lower, upper, step;

lower = 0;

upper = 300;

step = 20;

fahr = lower;

printf("\n\nCelsius Fahrengeit\n");

do

{

celsius = 5.0 \* (fahr - 32.0) / 9.0;

printf("%10.0f\t%8.1f\n", fahr, celsius);

fahr = fahr + step;

} while (fahr <= upper);

return 0;

}

4.

// Здійснити модифікацію цієї програми з оператором for:

/\* Celsius and Fahrengeit \*/

/\* C=(5/9)\*(F-32) \*/

#include <stdio.h>

// int main()

// {

// float fahr, celsius;

// int lower, upper, step;

// lower = 0;

// upper = 300;

// step = 20;

// printf("\n\nCelsius Fahrengeit\n");

// for (fahr = lower; fahr <= upper; fahr = fahr + step)

// {

// celsius = 5.0 \* (fahr - 32.0) / 9.0;

// printf("%10.0f\t%8.1f\n", fahr, celsius);

// }

// return 0;

// }

int main()

{

float fahr;

int lower, upper, step;

printf("\n\nCelsius Fahrengeit\n");

for (fahr = 0; fahr <= 300; fahr += 20)

{

printf("%10.0f\t%8.1f\n", fahr, 5.0 \* (fahr - 32.0) / 9.0);

}

return 0;

}

5.

// Модифікувати програму виведення парних чисел від 1000000 до 0 застосувавши в операторі циклу for операцію «кома».

#include <stdio.h>

int main()

{

for (int i = 1000000; i >= 0; i -= 2)

printf("%d\n", i);

return 0;

}

6.

// Обчислити скільки зерен необхідно було би видати винахідникові шахів, якщо за першу клітинуу шахівниці він попросив видати одну зернину пшениці, а за кожну наступну в двічі більше за попередні. У шахівниці 64 клітини.

#include <stdio.h>

int main()

{

long long int grains = 1;

for (int i = 1; i < 64; i++)

{

grains \*= 2;

}

printf("grains = %lld\n", grains);

return 0;

}

7.

// . Для цілих чисел від 1 до 20 обчислити квадратні, кубічні та корені четвертого порядку. Результати звести у таблицю, використовуючи форматування функції printf().

// Path: Chaikovski\lab5\lab5\_8.c

#include <stdio.h>

#include <math.h>

int main()

{

int i;

printf("i\ti^2\ti^3\ti^0.25\n");

for (i = 1; i <= 20; i++)

{

printf("%d\t%d\t%d\t%f\n", i, i \* i, i \* i \* i, pow(i, 0.25));

}

return 0;

}

8.

#include <stdio.h>

// Програма розв’язку рівнянь методом половинного ділення без застосування операторів циклу.

// Рівняння

// e – точність обчислення кореня, a,b – інтервал невизначеності.

// int main()

// {

// float a = -100, b = 100, e = 0.001, x, y, y1;

// x = a;

// y = x \* x - 81;

// y1 = y;

// m1:

// if ((b - a) < e)

// goto m2;

// x = (a + b) / 2;

// printf("x=%f\n", x);

// y = x \* x - 81;

// if (y \* y1 >= 0)

// {

// a = x;

// goto m1;

// };

// b = x;

// goto m1;

// m2:

// printf("korin'=%f", x);

// return 0;

// }

// Застосувавши оператори циклу, модифікувати програму розв’язку алгебраїчного рівняння методом половинно ділення, коли інтервал невизначеності а-b ділять навпіл і перевіряють в якій половині знаходить корінь. Ця половина стає новим інтервалом невизначеності і вся процедура повторюється до тих пір док не буде досягнута задана точність розрахунку.

// #include <stdio.h>

int main()

{

float a = -100, b = 100, e = 0.001, x, y, y1;

x = a;

y = x \* x - 81;

y1 = y;

while ((b - a) > e)

{

x = (a + b) / 2;

printf("x=%f\n", x);

y = x \* x - 81;

if (y \* y1 >= 0)

{

a = x;

}

else

{

b = x;

}

}

printf("korin'=%f", x);

return 0;

}

9.

// Здійснити табулювання функції, що з певними припущеннями з достатньою точністю моделює імпульс Максвела, який утворюється при ударному збудженні широкосмугової антени. Обчислення провести на проміжку зміни і в межах [0-31] з кроком i=1, N=32. Результати вивести у вигляді таблиці. Визначити найбільше та найменше значення функції на цьому проміжку.

// y = i ^ 2 \* e ^ ((-i^2)/100) \* sin (2\*pi/N\*i)

#include <stdio.h>

#include <math.h>

int main()

{

int i, N = 32;

double y;

printf("i\ty\n");

for (i = 0; i <= 31; i++)

{

y = pow(i, 2) \* exp((-pow(i, 2)) / 100) \* sin(2 \* M\_PI / N \* i);

printf("%d\t%f", i, y);

printf("\n");

}

return 0;

}

10.

#include <stdio.h>

#include <math.h>

int main() // Otsinka mashynnoho nulia

{

int i = 0; // і – лічильник ітерацій

float precision, a; // а- допоміжна змінна

precision = 1.0; // precision – обчислювана точність відносно числа 1.0

m:

precision = precision / 2.;

a = precision + 1.0;

i++;

if (a > 1.0)

goto m;

printf("\n число ділень на 2: %6d\n", i);

printf("машинний нуль: %e\n ", precision);

return 0;

}

// модифікувати програму застосувавши кожного разу один із трьох циклічних операторів. Оцінку ”машинного нуля” провести також для даних типу double - формат виведення %le, long double формат виведення %Le.

#include <stdio.h>

#include <math.h>

int main() // Otsinka mashynnoho nulia

{

int i = 0; // і – лічильник ітерацій

double precision, a; // а- допоміжна змінна

precision = 1.0; // precision – обчислювана точність відносно числа 1.0

m:

precision = precision / 2.;

a = precision + 1.0;

i++;

if (a > 1.0)

goto m;

printf("\n число ділень на 2: %6d\n", i);

printf("машинний нуль: %le\n ", precision);

return 0;

}

// Аналогічно з long double.

11.

// Заповнити екран монітора символами так, щоб утворити прямокутний трикутник зображений на рисунку, використавши для цього вкладені цикли.

// Символи для заповнення вибрати із таблиці ASCII згідно свого варіанту починаючи із коду, що дорівнює в десятковій системі 100 (символ d).

#include <stdio.h>

int main()

{

int i, j;

int symb;

printf("Your triangle:\n");

scanf("Ender num 0 to 100: %d", &symb);

for (i = 0; i < 10; i++)

{

for (j = 0; j < i; j++)

{

printf("%c", symb);

}

printf("\n");

}

}

12.

// Обчислити значення скінченої суми, або добутку згідно свого варіанту. Врахувати, що навіть для невеликих чисел значення факторіала може вийти за гранично допустимі для даного типу даних. Аргумент тригонометричних функцій задавати в межах: 0 ≤ x ≤ 2/π.

// Дано натуральні числа N і М=5. Обчислити

// F = (M! + N!) / (M + N)!

#include <stdio.h>

int factorial(int n)

{

int i;

int result = 1;

for (i = 1; i <= n; i++)

{

result \*= i;

}

return result;

}

int main()

{

int n, m;

printf("Enter n: ");

scanf("%d", &n);

printf("Enter m: ");

scanf("%d", &m);

printf("F = %d\n", factorial(m) + factorial(n) / (m + n));

return 0;

}

13.

// Відомо, що одним із методів обчислення багатьох функцій є розкладання їх в ряд Тейлора:

// Завдання: для заданого х, яке уводиться з клавіатури під час роботи програми,

// y = ch(x) = 1 + x^2/2! + x^4/4! + x^6/6! + ... + x^2n/(2n)!

// обчислити значення функції y за допомогою бібліотечних функцій компілятора так і за допомогою вище наведеного явного розкладу її в ряд (ітераційний процес до досягнення заданої точності). Обчислити при цьому також кількість ітерацій або кількість членів ряду в розкладі функції. Точність обчислень, тобто значення члена ряду розкладу функції коли необхідно припиняти ітераційний процес, a=0.00001. Аргумент тригонометричних функцій задавати в межах: 0 ≤ x ≤ 2/π.

#include <stdio.h>

#include <math.h>

int main()

{

// Example Enter x: 0.5

double x, y, a = 0.00001;

int n = 0;

printf("Enter x: ");

scanf("%lf", &x);

y = cosh(x);

printf("y = %lf\n", y);

y = 1;

double y1 = 1;

double y2 = 1;

while (fabs(y1) > a)

{

y1 \*= x \* x / (2 \* n + 2) / (2 \* n + 1);

y += y1;

n++;

}

printf("y = %lf\n", y);

printf("n = %d\n", n);

return 0;

}

Висновок: на даній лабораторній роботі я дослідив властивості циклічних операторів мови С.